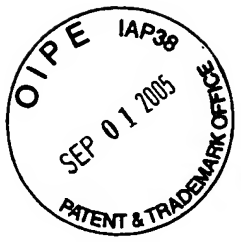


Attorney Docket No. 06502.0383-00
U.S. Application No. 10/021,084

PATENT
Customer No. 22,852
Attorney Docket No. 06502.0383-00

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of:	)	
	)	
Lewis CURTIS et al.	)	Group Art Unit: 3629
	)	
Application No.: 10/021,084	)	Examiner: THAI, Cang G
	)	
Filed: December 19, 2001	)	
	)	
For: SYSTEMS AND METHODS FOR PROVIDING A CUSTOMER RELATIONSHIP MANAGEMENT ARCHITECTURE	)	Confirmation No.: 8738

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

DECLARATION OF JOHN CRUPI UNDER 37 C.F.R. § 1.131
EXHIBIT 1

Technical Report (15 total pages).

[redacted]

Timedisplay : 24 hour format Timezone : GMT-7 (US/M

[redacted]

Name : John Crupi [redacted]

Executive Summary

The Net Economy has changed the rules for Customer Relationship Management (CRM) solutions, yet most legacy CRM architectures do not readily adapt to change. For many companies, there is now a pressing need to implement a more agile, Web–centric "eCRM" architecture that delivers on a new set of customer requirements while meeting increasingly stringent business objectives.

With its cross–platform capabilities, Web–centric functionality, and broad industry adoption, the Java 2 Enterprise Edition (J2EE) platform provides an excellent foundation for building a complete, customized, standards–based eCRM architecture.

This paper presents the core concepts, strategies, and technologies needed to build an agile architecture for deploying eCRM solutions. It summarizes the new market realities that impact architectural design, describes the J2EE platform as the basis of an effective eCRM architecture, and provides a real–world example to illustrate key points. The paper also shows you how a J2EE compliant architectural framework can provide a bridge between your legacy infrastructure and a more flexible, scalable, reliable environment.

As an industry leader in network computing and Internet technologies, Sun is in a unique position to help you examine the opportunities, consider the issues, and rapidly implement complete and effective eCRM architectural solutions.

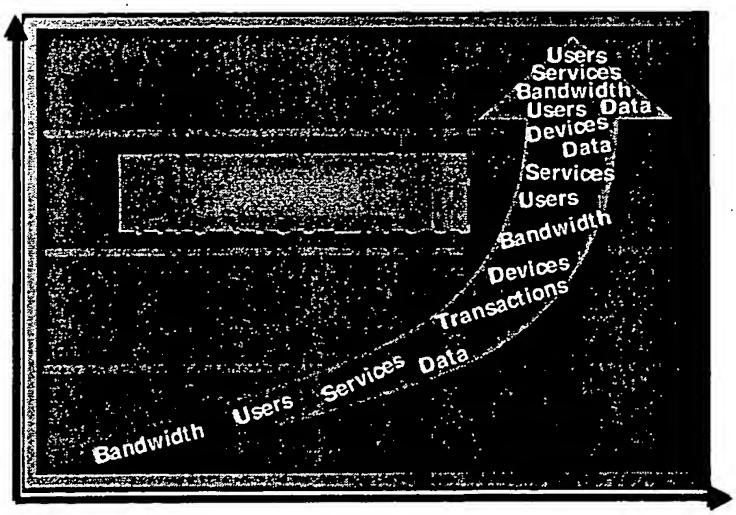
eCRM: New Realities, New Rules

Traditional CRM solutions focused on improving operational efficiencies of a company's relationship with its customers. And they accomplished this objective. However, the dramatic growth in Web–based commerce and communications has created new market realities and new technical requirements that legacy CRM architectures simply weren't designed to address. The new discipline, eCRM, requires a Web–centric approach to synchronizing customer relationships across communication channels, business functions, and companies. This section examines the impact on eCRM architecture.

The "Net Effect" Creates New Customer Requirements

The Net Effect is a phenomenon characterized by the explosive––and simultaneous––growth of everything related to network computing: the number of users, the diversity of devices, the amount of bandwidth, the transaction volume, the quantity of network services, and the volume of data. Each of these elements is growing at a tremendous rate on its own. When combined into a single measurement of the growth of network computing, the result is an exponential growth curve.

The Net Effect also creates an upward spiral in the *value* of the network, because the more powerful the network becomes the more it is used, and the more it is used the more important it becomes as a business tool. The Net Effect also creates a dramatic increase in consumer power, because it makes the network a focal point for competitive differentiation.



The Net Effect is the cumulative result of the explosive growth in bandwidth, number of users, types of network services, volume of data, quantity of transactions, and diversity of devices.

As a result, the Net Effect magnifies the importance of systemic qualities such as service–level availability, reliability, scalability, manageability, and security, requiring an architecture that can deliver always–on, easily accessible, high–quality service delivery. At the same time, customers are now expecting—and demanding—new levels of service and network performance that your eCRM solution must address, including:

- The ability to choose among multiple channels for access into your company, including Internet channels such as e–mail and Web and wireless channels
- Instant, consistent response from all channels and functions
- Reliable assistance from any company employee on virtually any question, including product information, ordering, problem tracking, or any other service related to the transaction
- Personalization of all interactions

To the IT department, these new market realities also create new business requirements: to deliver eCRM solutions in "Internet time," provide the flexibility to change features quickly, and offer unparalleled quality of service (QoS) to customers.

Limitations of Traditional CRM Technical Architecture

Traditional CRM solutions aimed at creating an integrated, enterprise–wide view of the customer, and therefore the technical architecture focused on linking islands of customer data and business processes. Typically, CRM vendors built their solutions based on the two–tiered client–server model. However, the advent of the Internet and new customer requirements soon exposed the flaws of traditional CRM models. Among the limitations:

- The two–tiered client–server model does not provide adequate scalability for today's unpredictable service–delivery workloads.
- The enhanced three–tiered client–server model, which provided additional flexibility through separation of business logic from system services, required the purchase of supported (proprietary) operating systems, Web browsers, and hardware, which created vendor lock–in.
- Web–enabling the client–server model does little to change the inflexibility of the underlying client–server architecture. While Web enablement achieves the short–term objective of making diverse systems available over the Web, the resulting Web presence is fractured and difficult for customers to navigate. More seriously, this approach makes it extremely difficult to combine the functionality of various silos and create new value–added services.

Design Principles for Web–Centric eCRM Architecture

To meet the new requirements of the Internet age, the eCRM architecture needs to bridge the companies disparate data silos and processes to allow for an integrated, Web–centric presentation of your business. To meet this challenge, your architecture will need to adhere to several overarching principles. Your technical architecture should be:

- **Physically distributed**: It should be possible to distribute processing across multiple physical network devices. This maximizes your ability to protect (through the firewall) discreet processing elements and to meet spikes in demand for services using horizontal scaling.

- **Logically tiered for separation of concerns**: Functionality should be partitioned according to logical function, enabling services to evolve independently of one another and meet new requirements individually.

- **Services–based, not code–based**: The unit of re–use should not be code; functionality should be deployed as components that are readily available across the enterprise and capable of being managed and controlled to achieve the needed quality–of–service (QoS) levels.

- **Assembled, not built**: Using services as building blocks, systems should be formed by assembling existing services, not by starting from scratch. Some of the parts will be provided outside of the corporate boundaries. Different types of systems will assemble things differently; for example, portals assemble content and integrate fragments that are provided by various services, and legacy applications are accessed through adapter wrappers.

- **Implemented in layers**: Layering maximizes the independence of processing components from their underlying platform implementations, providing maximum flexibility in selecting, tuning, and evolving platforms in response to changes in demand or technologies.

- **Externally managed and controlled**: Services need to participate in a coherent management and security framework, established as a centralized facility—as opposed to implementing these capabilities in the components themselves, resulting in a menagerie of mechanisms and interfaces.

Services–Driven Architecture

The architectural goal for eCRM solutions is to build a services–driven architecture—enabling rapid development and flexible deployment of new Web–centric offerings. Sun's approach to creating service–driven architecture is to add a "services layer" between client devices and back–end resources. Through this approach, back–end resource functionality is made available via well–defined, network–callable components. These components can make back–end functionality readily available to Web clients, but unlike the silo approach they provide building blocks for combining simple services into full–featured, value–added services that themselves become Web–available.

Creating this architecture involves standardizing interface technology such that once deployed, components are available for use by other components without complex message reformatting or protocol conversions. The enabling technology for this standardization is the J2EE software platform.

J2EE: Cornerstone of an Adaptable, Web-Centric eCRM Architecture

The J2EE platform is a unified software foundation that simplifies multi-tier enterprise application development, enabling you to protect and leverage your existing infrastructure investments. By integrating industry-tested and widely adopted enterprise Java technologies — such as Enterprise JavaBeans technology and JavaServerPages technology -- and by providing support for the eXtensible Markup Language (XML), J2EE provides an open, end-end solution upon which to build end-to-end e-commerce solutions.

Commercial software products designed to the J2EE architecture standard are flexible enough to allow use of almost any platform (servers and clients) for deployment, while offering segmented tiers of functional control and management. Furthermore, this approach enables the system to be managed by any J2EE compliant application server on the market—so you can manage the complexities of scalability, reliability, availability, manageability, and secureability required by customers.

To date, more than 50 industry leaders have endorsed the J2EE platform. The strong industry adoption of J2EE software provides a robust marketplace from which you can choose the application that best meets your particular needs.

The Role of J2EE in eCRM Architecture

The J2EE platform defines a standards-based component architecture supporting multi-channel, service-driven applications. Key components of the specification include:

- The Servlet and Java Server Pages specifications, which standardize Web server plug-ins, and template-driven HTML generation
- The Enterprise Java Beans (EJB) specification, which provides a standard abstraction for application servers
- The Java Enterprise APIs such as Java Database Connectivity (JDBC), Java Messaging Services (JMS), and Java Naming and Directory Interface (JNDI), which standardize the interfaces to databases, message-oriented middleware, and directory services

The J2EE N–tier Application Model

The N–tier (or multi–tier) architectural approach enabled by J2EE segregates business logic from the presentation and data layers, enabling organizations to propagate changes in business rules immediately throughout the organization. Because business logic remains independent of access channels and resource implementations, the enterprise gains the flexibility to combine different access methods and back–end resource technologies. With application logic residing on separate servers, organizations can vertically or horizontally scale each tier to support growing numbers of users and higher volumes of traffic. Another key advantage of a functionally partitioned architecture is improved response times for accessing data via the Web. In addition, virtual platform layering allows architects and developers to spend more time focusing on solving business problems rather than technical issues.

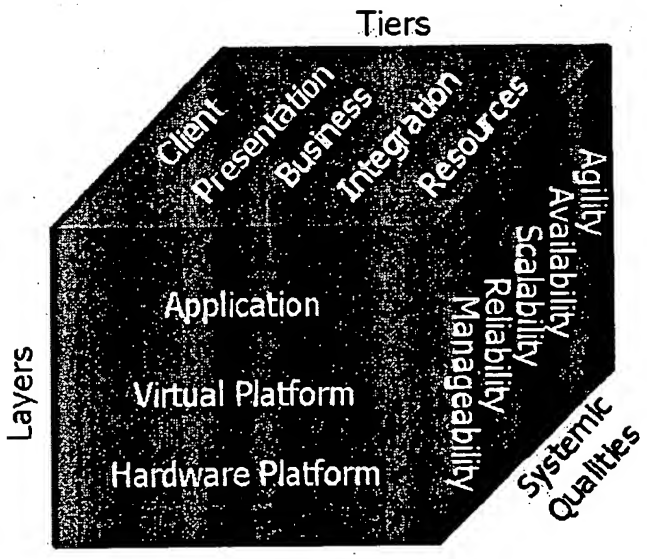
There are three key concerns or "dimensions" to be addressed in building a service–driven eCRM architecture:

- ***Tiers***: The logical partitioning of functionality with well–defined relationships to other services (functional separation of concerns and topology).

- ***Layers***: The hardware and software stack that hosts services within a given tier; physical, network, and software platforms and standard API sets that support your components which provide a service.

- ***Systemic Qualities***: The strategies, tools, and practices that will deliver the requisite quality of service (e.g. availability, scalability, security, and manageability).

These three dimensions have orthogonal relationships: for example, each systemic quality needs to be provided in each tier; each tier may have different optimal choices of implementations or features in its platform layers; and each layer may enable different strategies for a given capability, which may be more or less appropriate for each tier.



By separating these concerns and relating them in this way, the tiers, layers, and operational capabilities define a design space onto which you can map specific products, tools, and application components, and against which you can gauge the completeness of the overall systems architecture. The next sections describe each of the dimensions of in more detail.

Application Tiers

By implementing a properly tiered architecture, an enterprise can take advantage of any new delivery channel regardless of the associated access technology, and can remain independent of back–end resource implementations. Sun recommends a five–tier model that includes the following tiers:

- ***Client services***: Reside on the client device or system and manage display and local interaction processing.

- ***Presentation services***: Aggregate and personalize content and services into channel–specific user interfaces. This entails the assembly of content, formatting, conversions, and content transformations—anything that has to do with the presentation of information to end users or external systems.

- ***Business services***: Execute business logic and manage transactions. Examples range from low–level services such as authentication and mail transport to true line–of–business services such as order entry, customer profile, payment, and inventory management.

- *Integration services*: Abstract and provide access to external resources. Due to the varied and external nature of these resources, this tier often employs loosely coupled paradigms such as queuing, publish/subscribe communications, and synchronous and asynchronous point–to–point messaging.

- *Resources*: Includes legacy systems, databases, external data feeds, specialized hardware devices such as telco switches or factory automation, and so on.

Platform Layering

Layering enables you to separate processing components from their underlying platform implementations, giving you maximum flexibility in selecting, tuning, and evolving technologies for your architecture. As shown in the "cube" diagram above, Sun interposes a virtual platform layer between the application layer and the hardware platform layer. The virtual platform layer provides standard APIs and specifications for products such as Web servers, application servers, and various types of middleware. By writing applications so that they depend only on the virtual platform APIs, developers can make applications portable across upper platform products.

To put it simply, if you choose your APIs wisely at the virtual platform level, the payoff is platform independence. And if you adhere to the standard APIs in the application layer products you deploy on your network––such as ERP, CRM, supply chain management, or sales force automation–– these applications will also be platform–independent.

Systemic Qualities

In the past, service–level requirements such as scalability and reliability were afterthoughts. Now with system demands expanding by orders of magnitude, these requirements have become strategic mandates that must be embedded both in the application architecture and in the development process. Among the new demands for systemic qualities today:

- *Reliability:* When customers spend a great of money on enterprise CRM solutions, they want a reliable application. Applications built upon the standard APIs of the J2EE platform are constantly tested, increasing reliability. While the largest CRM ISV vendor might advertise 1000 deployments in its lifetime (all with different versions and different proprietary API components), thousands of applications built around the J2EE standard API are deployed each month.

- *Scalability:* In a Web–centric world, massive scalability is critical to support unpredictable surges in demand for network services. Sun's philosophy is that scaling should not require changes in the architecture or the loss of services during the time needed to scale. If your eCRM solution is based on J2EE, it can run on any J2EE compliant application server, while supporting both vertical and horizontal scalability and session management.

- *Manageability* Customers demand a clear, coherent management model that separates tier logic utilizing 'best of breed' component solutions to deliver eCRM functionality. While previous management technologies were often proprietary and constrained functionality, the J2EE platform enables you to select best–of–breed components and integrate them into a comprehensive management solution.

- *Agility:* No two companies are alike, so eCRM solutions need to be configured to meet the unique needs of each enterprise. Without standards, everyone takes a different approach to customization, choosing a different way to update tables and a different fourth–generation language tool. J2EE is the middle ground between a rigid packaged solution and a complete custom–built application. It enables vendors to deliver exactly the functionality that a specific enterprise needs without having to construct the underlying infrastructure to support it. In addition, Java enables applications to be developed, updated and customized much faster and more efficiently, because the supply of Java developers is high compared to the number of specific CRM ISV vendor programming developers.

- *Availability:* Traditional CRM vendor solutions provided high availability through the hardware/operating systems layer. However, this did not always ensure high service–level availability, and many other variables can impact the uptime of the application. The J2EE platform supports both stateless and stateful EJB processing environments. In a call center environment, for example, the server running in a stateless session creates an object through class instantiation, processes the specific transaction object in memory and drops the object when the transaction is completed. If the server goes down in the middle of the transaction, this can have major ramifications, requiring that the customer service representative to restart the transaction (sometimes through re–login and re–enter of unsaved database information) and application server must recreate the object. However, J2EE also can also support stateful sessions, allowing the specific transaction to be clustered among multiple servers in a high–availability, real–time system. Traditional CRM solutions have only supported stateless transaction sessions.

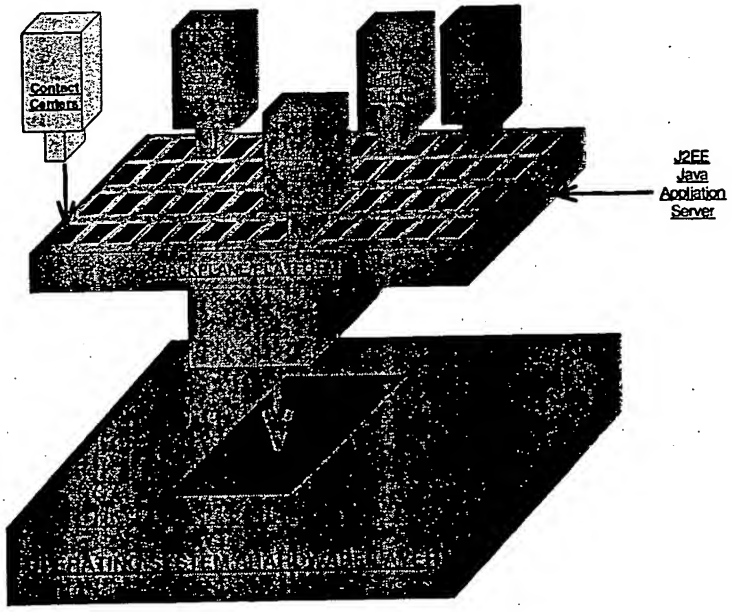
J2EE Containers and Contracts

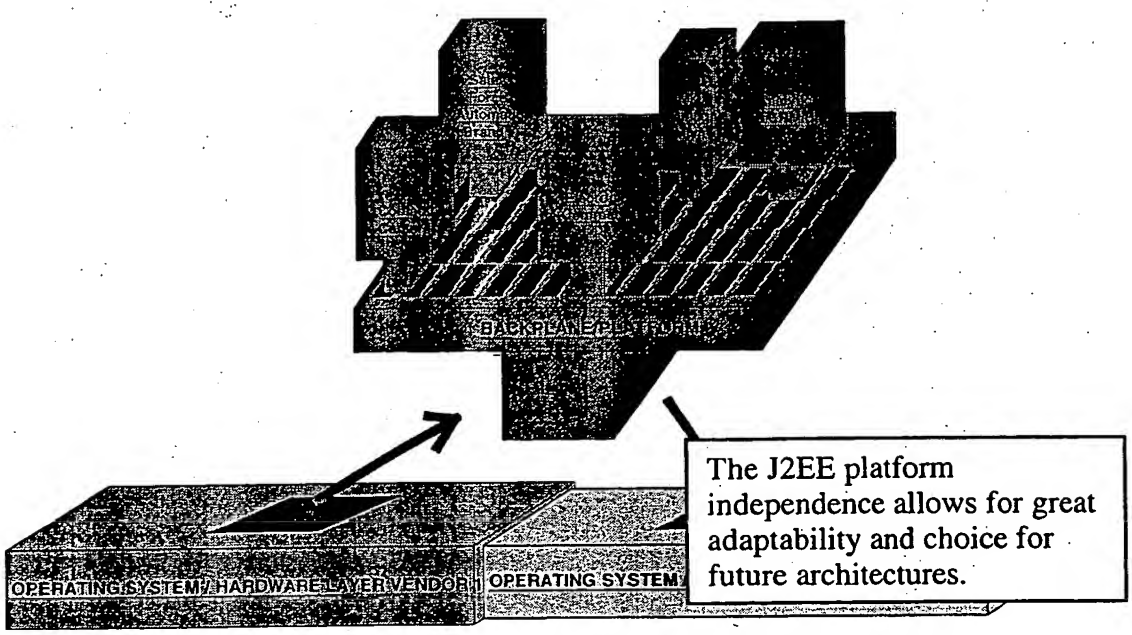
Traditional CRM architectures used proprietary designs to run on specific platforms with a specific number of processors and nodes. As the technical requirements changed, customers often had to purchase a newer version of the system or a completely different system. The J2EE platform addresses these issues with the concept of *container* and *contracts*. The container is a kind of "virtual plugboard" into which a particular type of component can be integrated. There are several types of containers, each operating in one of the four tiers.

Currently, J2EE defines the EJB container, into which EJBs plug, and the Web container, into which Servlets and JSPs plug. The "plugs" are actually APIs that the component must implement. These APIs form the *Component Contract* between the component and its container. The container must expose the services of the component to its clients via a set of APIs, so that the client can plug into the container. Finally, the application server vendor must implement the container in accordance with the industry specification for the specific Component/container

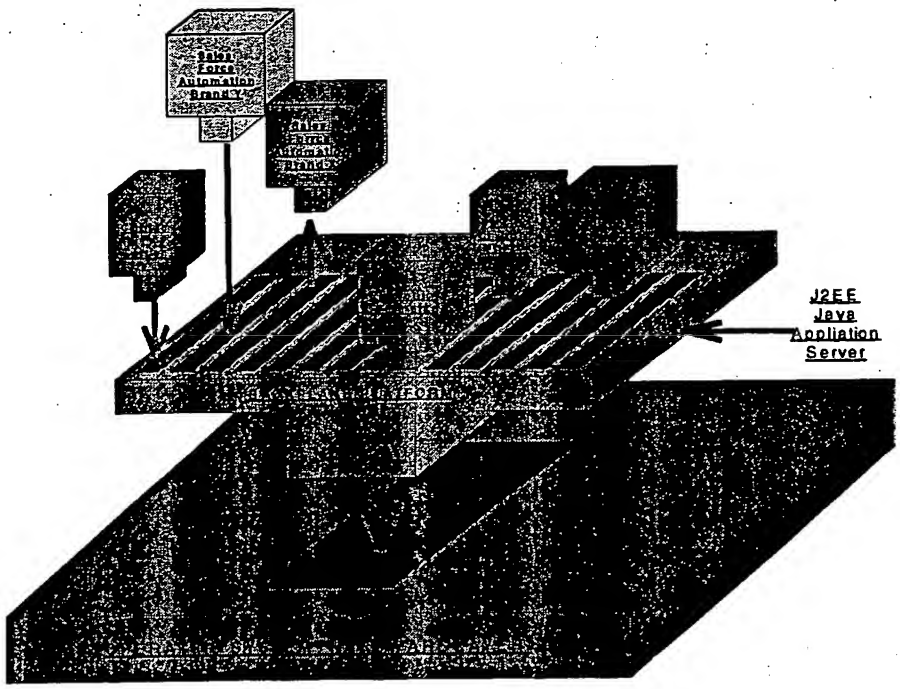
type. In this manner, J2EE provides a variety of component and container types including Enterprise Java Beans, Servlets, Java Server Pages, Applets and Application clients. Client contracts into these containers are supported by various connectivity technologies including JNDI, RMI, RMI/IIOP, IIOP, JMS, JTA, JDBC and J2EE Connectors.

Thus, these component types, their containers and contracts provide multi–tier enterprise architecture for interchangeable components that can be supplied by an open marketplace. In turn, these containers are implemented by application server products that are also supplied on an open marketplace. These server products can differentiate themselves by various levels of quality of service – and price points.





The J2EE platform independence allows for great adaptability and choice for future architectures.



A J2EE Functional Solution can be replaced without replacing the entire architecture.

Benefits

The J2EE compliant architectural model for eCRM described in this document offers the following benefits for businesses, end users, suppliers, and customers:

- ***Internet Time deployment and delivery***: It is easier to identify hot spots and problems in the network, such as overloaded services or security issues, and to increase system support or make modifications to specific services. Also, since services are highly re–useable, it is easier to deploy new services quickly or to build on existing services.

- ***Tuned deployment***: The J2EE compliant framework enables you to set the desired levels of scalability, availability, security, and bandwidth consumption individually. This allows you to match system–level resources to services.

- ***Outsourcing opportunities***: Since network services are based on well–defined interfaces and are accessible from anywhere on the network, you now have the option of outsourcing network services to third–party services providers.

- ***More efficient use of resources***: With J2EE compliant architecture, you can pool resources and use what you have more efficiently. When you pool resources to create a service, you can aggregate free resources and make full use of them.

- ***Flexible use and re–use of services***: Since services are available over the network they can be used in numerous ways. And since the services are available to the network, every time you add new services you are enriching the entire network environment.

- ***Architectural consistency***: Based on standards, the J2EE compliant architecture is easier to architect than traditional networks and makes it easier to "see the big picture" and keep track of the environment.

- ***Modularity:*** In a highly networked environment, the IT department needs to be able to work on individual parts of the network without having to understand the entire architecture. The J2EE compliant architecture removes the need to know how each is deployed.

Real–World Example:
Fortune 100 Company Jumpstarts its e–Commerce Web Site with Java and XML

Like many established brick and mortar companies who are scrambling to build their e–business strategies and implement them in "Internet time," one major manufacturer has decided to revamp its e–commerce site for greater flexibility, adaptability and scalability for global market growth. To accomplish its objective for superior performance in a global environment, this company realized its first order of business was to develop its B–to–C e–commerce infrastructure on a scalable, high performing, and easily extensible architecture. Currently, the manufacturer's e–commerce site receives more than 3 million hits per day and plans to expand its reach into over 30 countries over the next 12 months.

The manufacturer's current site took approximately two years to build and required an initial investment of over $10 million. As the company has needed to make modifications to the site's HTML pages, it has invested significant incremental dollars due to a proprietary, non–scalable platform. Frustrated by this, and seeking a way to add business rules and personalization to its site, the manufacturer turned to RedCelsius's e–Commerce and eCRM platform.

As an enterprise–class, J2EE–compliant set of applications that incorporates state of the art technology including an n–tier architectural approach utilizing Enterprise Java Beans, Java Server Pages, Java Beans, Java Servlets and XML, the e–Business Suite will provide the company with a truly scalable, robust platform. Because of the platform–independent features of Java and J2EE, the company now has tremendous flexibility. As growth and performance needs dictate, migration to more robust hardware, database and operating system platforms will be easily supported. Support for various client devices beyond the browser are also easily supported by the e–Business Suite; for example, support for user interactions via kiosks or wireless devices could be added.

As the manufacturer expands its client base into new countries, the Java compliant technology infrastructure with underlying data will be replicated for different languages and will support an extensive and growing array of functional and data requirements for varying cultural preferences and legal requirements. RedCelsius' J2EE component architecture, as well as a separation between the application, web and client tiers, and the use of XML and XSL will provide the flexibility needed to support the creation of any type of application.

The use of XML and XSL provides flexibility and support for rendering the presentation of these components in a manner consistent with the look and feel of the organization's web site. RedCelsius' data–driven architectural capabilities and use of meta–data support the persistence of any type of registration data. Finally, RedCelsius' Enterprise Application Integration (EAI) layer and the use of XML supports integration with any corporate enterprise system.

How Sun Can Help

Many of the architectural concepts presented in this paper, such as service tier processing, platform layering, and containers, are new to the IT organization and programmers. With expertise in Java technology and developing and deploying J2EE compliant architectures for eCRM, the Sun Professional Services organization can serve as a trusted advisor or active participant in any aspect of your efforts to design and build a more agile architecture for your next–generation eCRM solution.

For More Information

For additional information about any of the concepts, products, or services described in this document, please contact the Sun Professional Services organization, or visit our Web site at http://www.sun.com/sunps.

About Sun Microsystems, Inc.

Since its inception in 1982, a singular vision, "The Network Is The Computer" has propelled Sun Microsystems, Inc. to its position as a leading provider of high–quality hardware, software, and services for establishing enterprise–wide intranets and expanding the power of the Internet. With more than $15.7 billion in annual revenues, Sun can be found in more than 150 countries and on the World Wide Web at http://sun.com

###

This Page is Inserted by IFW Indexing and Scanning Operations and is not part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ BLACK BORDERS
- ☐ IMAGE CUT OFF AT TOP, BOTTOM OR SIDES
- ☐ FADED TEXT OR DRAWING
- ☐ BLURRED OR ILLEGIBLE TEXT OR DRAWING
- ☐ SKEWED/SLANTED IMAGES
- ☐ COLOR OR BLACK AND WHITE PHOTOGRAPHS
- ☐ GRAY SCALE DOCUMENTS
- ☑ LINES OR MARKS ON ORIGINAL DOCUMENT
- ☐ REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY
- ☐ OTHER: _____

IMAGES ARE BEST AVAILABLE COPY.
As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.